### 1. Introduction

**1.1 Product Overview**

We have developed an agent-based intelligent shopping system that uses multi-agent collaboration to meet personalized shopping needs. The system includes a human-computer interaction agent and multiple functional agents that automatically break down user shopping requirements (such as budget, preferences, and purpose) into subtasks. These subtasks are handled by e-commerce agents (such as Amazon) for queries, and the results are integrated to generate a shopping plan. The core features of the system are modularity and scalability, supporting dynamic adjustments and rapid iterations for optimization.

**1.2 Purpose of the Manual**

This manual aims to help users fully understand and efficiently use the intelligent shopping system. Through detailed guidance and instructions, users can easily get started and make the most of the system's features to meet personalized shopping needs. The manual is suitable for all users who wish to understand the system's functions and operations, including first-time users and advanced users seeking in-depth knowledge of the system's characteristics.

**1.3 Project Background**

The shopping system is based on the Mofa (<https://github.com/moxin-org/mofa>) intelligent agent software framework and developed by the SIMPLEST team as an intelligent shopping system that works through multi-agent collaboration. MoFa is a software framework for building AI agents using a compositional approach. Using MoFa, agents can be constructed through templates and combined in layers to form more powerful super agents, achieving enhanced functionality.

### 2. Software Overview

**2.1 Objectives**

**Automated Requirement Analysis**: Extract key information such as shopping budget, product preferences, and usage scenarios from user text input.

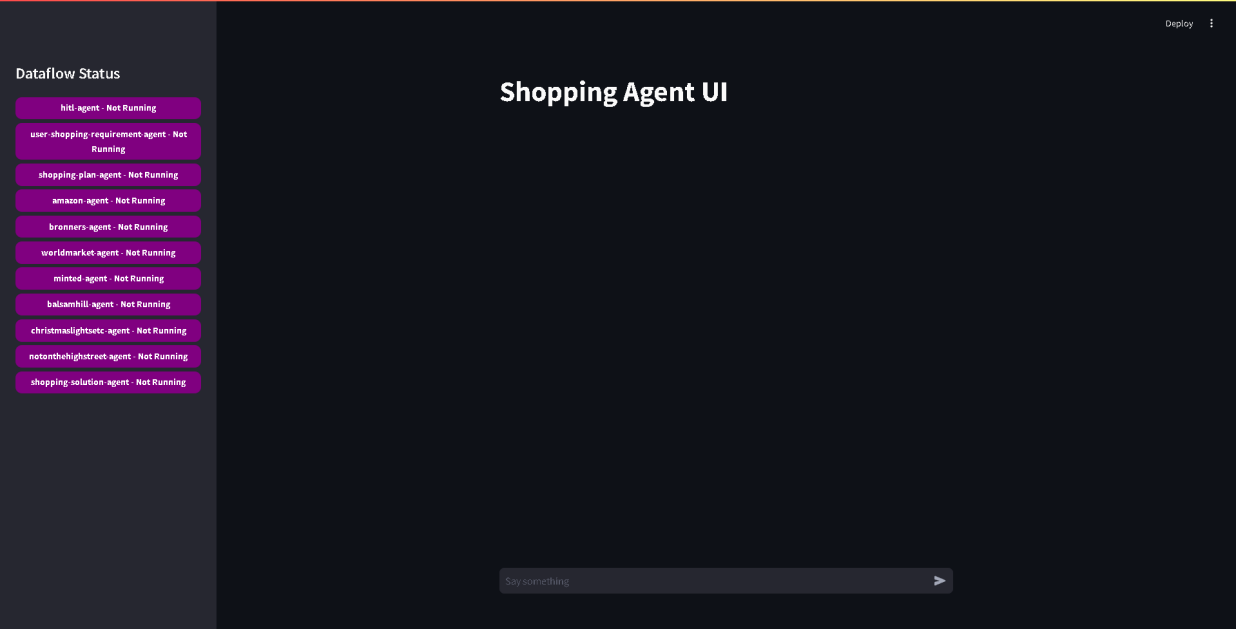
**Task Decomposition and Planning**: Automatically generate a list of query tasks based on user needs (e.g., search instructions for multiple e-commerce platforms).

**Multi-source Information Aggregation**: Obtain product information (price, specifications, stock, etc.) from different e-commerce agents, and perform cross-platform comparison and filtering.

**Plan Generation and Feedback**: Provide users with candidate shopping plans and support user feedback and further adjustments.

**2.2 Features**

**2.2.1 Front-end Features**



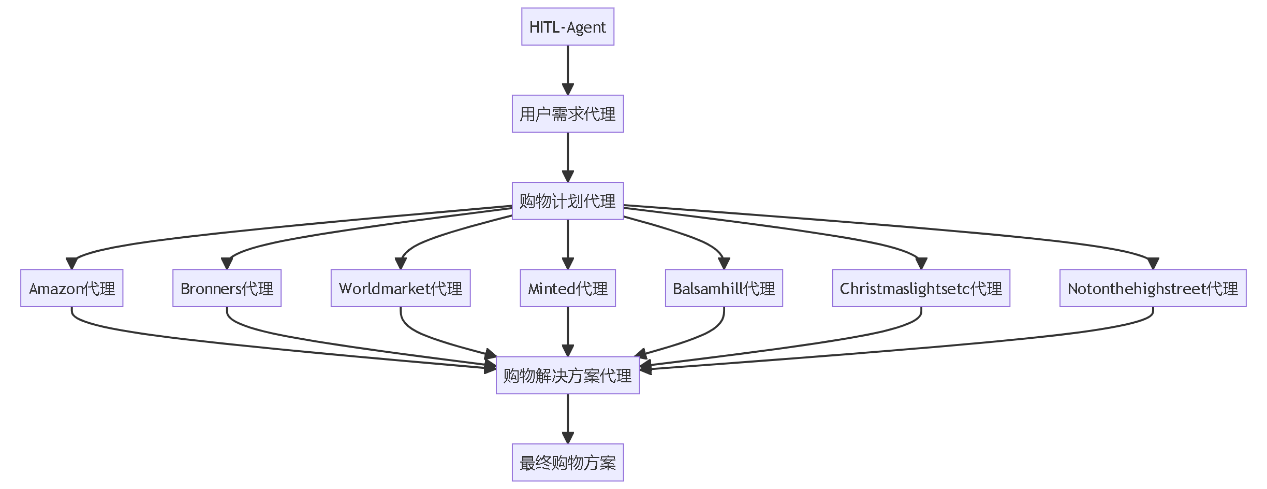
**Dataflow Status Panel**: Displays the operational status of various agents. Users can check whether each agent (e.g., HITL, Amazon, Bronners) is running, making it easy to monitor the overall system status. Note: Due to competition time constraints, there may be inaccuracies or delays in agent status updates. These will be addressed and improved in future versions.

**Chat Window**: Users can enter shopping needs and preferences here. The system will respond through the human-computer interaction agent and provide personalized shopping suggestions.

**Real-time Feedback**: The interface shows tasks currently being processed (e.g., “Bot is typing...”), allowing users to understand the current progress.

**Input Box**: Users can enter new commands or adjust preferences at any time in the input box at the bottom. The system will adjust its response strategies and provide more tailored feedback based on the input.

**2.2.2 Back-end Features**



a. **HITL-Agent (hitl-agent)**

Role: Entry point for user-system interaction.

Function: Receives initial user needs (user\_input) and feedback on plans (shopping\_solution\_user\_input, shopping\_plan\_user\_input), such as "prefer NVIDIA graphics card" or "increase budget to 8500 yuan."

b. **User Shopping Requirement Agent (user-shopping-requirement-agent)**

Function: Parses vague, natural language user needs into structured data for subsequent planning.

c. **Shopping Plan Agent (shopping-plan-agent)**

Function: Breaks down tasks into multiple e-commerce search instructions based on user needs, providing clear query parameters for each platform agent (e.g., search price range, brand, product type).

d. **E-commerce Agents** (e.g., amazon-agent, bronners-agent, worldmarket-agent, minted-agent, balsamhill-agent, christmaslightsetc-agent, notonthehighstreet-agent)

Function: Perform product queries and data retrieval for specific platforms, providing candidate product data for plan generation.

e. **Shopping Solution Agent (shopping-solution-agent)**

Function: Filters, combines, and ranks candidate products based on user needs and data returned from each platform, generating the final shopping plan.

The core processing logic involves modular task decomposition and execution, achieving fast, personalized shopping plan generation through parallel processing by multiple agents.

### 3. Operating Environment

**3.1 Supported Software**

a. **Operating System**:

Windows with WSL or Linux (Ubuntu 20.04 and above) and macOS .

b. **Programming Language and Compilation System**:

Python 3.10 and above.

c. **Other Necessary Software**:

Chrome and Chromedriver

To install necessary components, run:

sudo apt-get update

sudo apt-get install -y curl unzip xvfb libxi6 libgconf-2-4

wget https://dl.google.com/linux/direct/google-chrome-stable\_current\_amd64.deb

sudo apt install ./google-chrome-stable\_current\_amd64.deb

**Ensure Chrome is installed successfully:**

google-chrome --version

Get the Chromedriver version that matches the installed Chrome version from:

<https://chromedriver.chromium.org/>

Download, unzip, and configure the environment:

# Replace with the correct version number for your installed browser

wget https://chromedriver.storage.googleapis.com/86.0.4240.22/chromedriver\_linux64.zip

unzip chromedriver\_linux64.zip

sudo mv chromedriver /usr/bin/chromedriver

sudo chown root:root /usr/bin/chromedriver

sudo chmod +x /usr/bin/chromedriver

**Ensure Chromedriver is installed successfully:**

chromedriver --version

If you have previously installed Chromedriver on Windows and added its path to PATH, ensure it now points to the newly extracted Chromedriver.

图片 1

For more detailed steps, please refer to the deployment documentation in the source code repository.

For Mac Users: Please refer to Install ChromeDriver on Mac for detailed instructions.

“<https://www.swtestacademy.com/install-chrome-driver-on-mac/>"

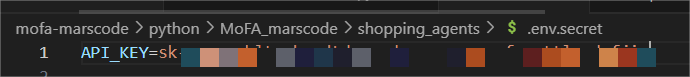
### 4. Quick Start

**4.1 System Initialization**

**4.1.1 Start the Backend**

Make sure to run the files in the order described below. Incorrect startup sequence will result in failure to run.

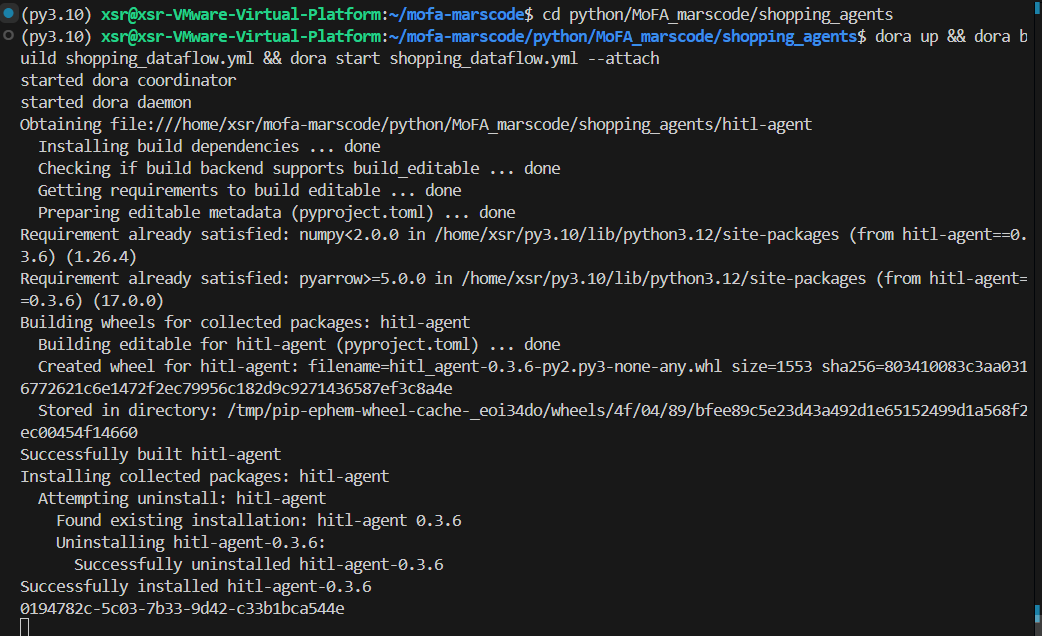
a. First, navigate to the python/MoFA\_marscode/shopping\_agents directory.  
Create a file named .env.secret in the current directory with the following structure:



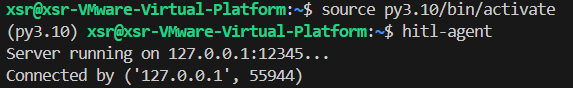
b. Run the following command in the current directory:

dora up && dora build shopping\_dataflow.yml && dora start shopping\_dataflow.yml --attach

If you see the ID number in the red box as shown in the image, it means the dataflow has started successfully.



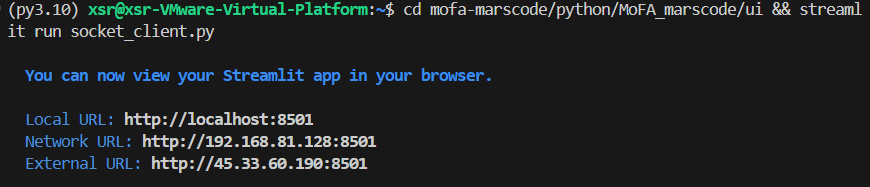
c. In another command terminal, run the hitl-agent.



d. Open a third command terminal and run the following command:

cd mofa-marscode/python/MoFA\_marscode/ui && streamlit run socket\_client.py

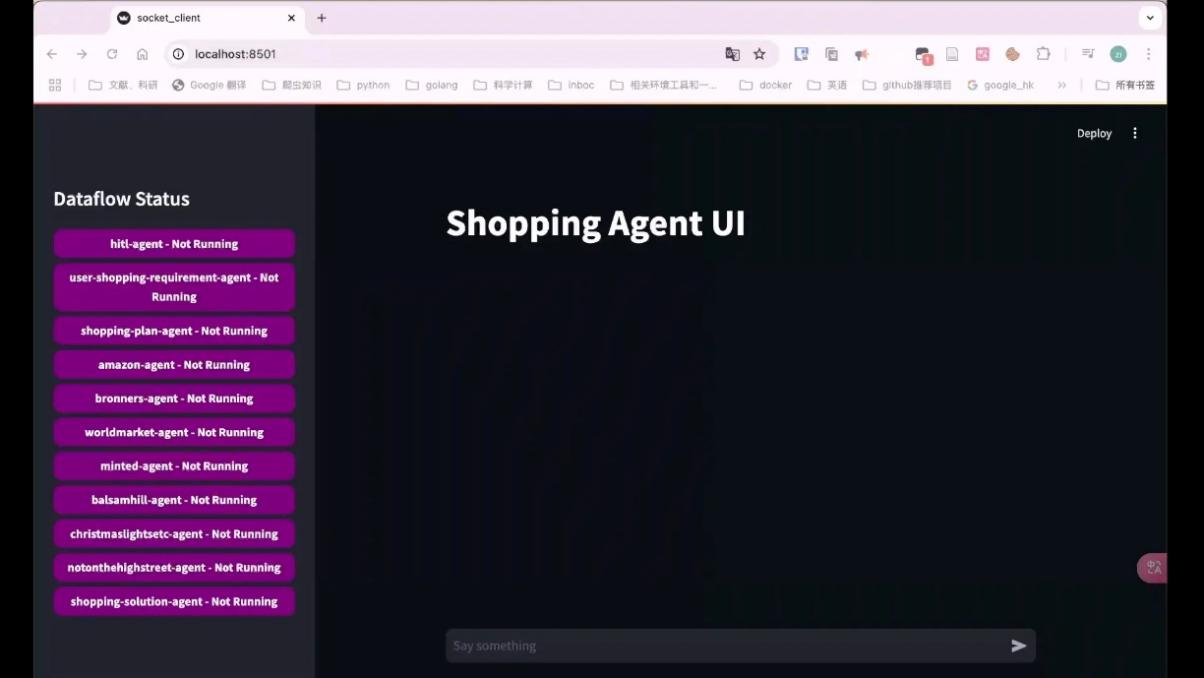
This will open your page.



Ensure that your port 12345 is not occupied. If it is, use lsof -i :12345 to check the process ID occupying it, and use kill -9 to terminate it.

**4.2.2 Front-end Usage**

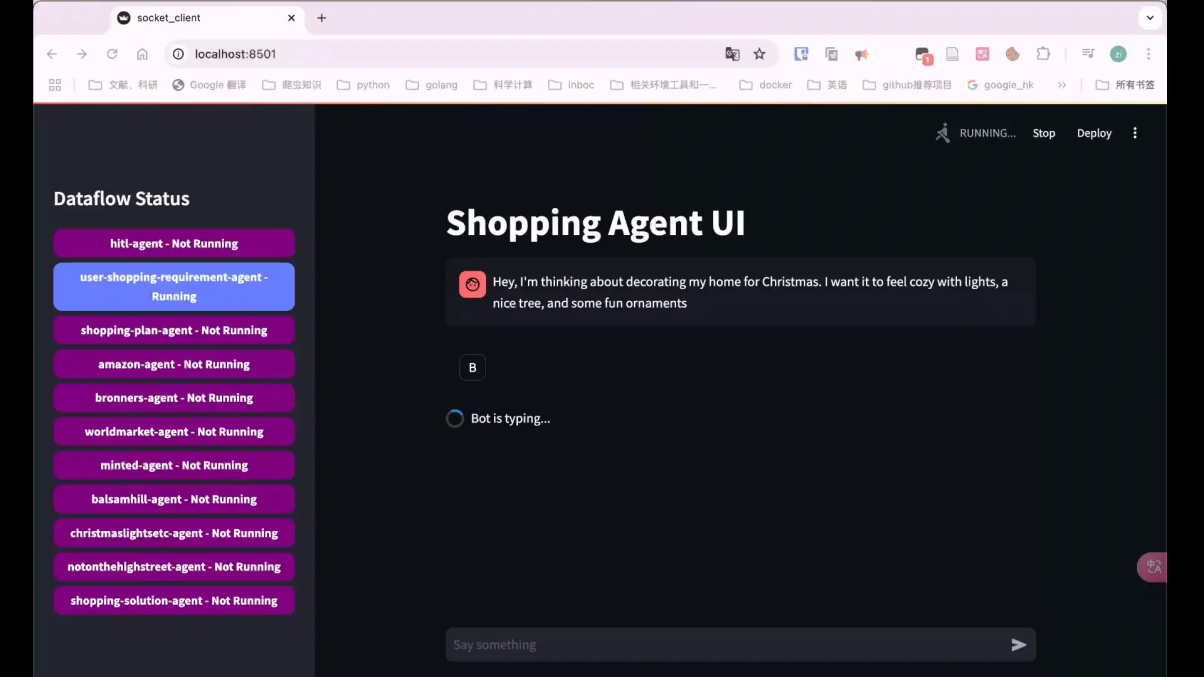
We will use "setting up a Christmas scene with decorations" as an example to demonstrate the process.



If you successfully start the backend, it will automatically redirect to the webpage shown above. Next, begin your dialogue task in the conversation box on the webpage.

**Step 1: Express Your Needs**

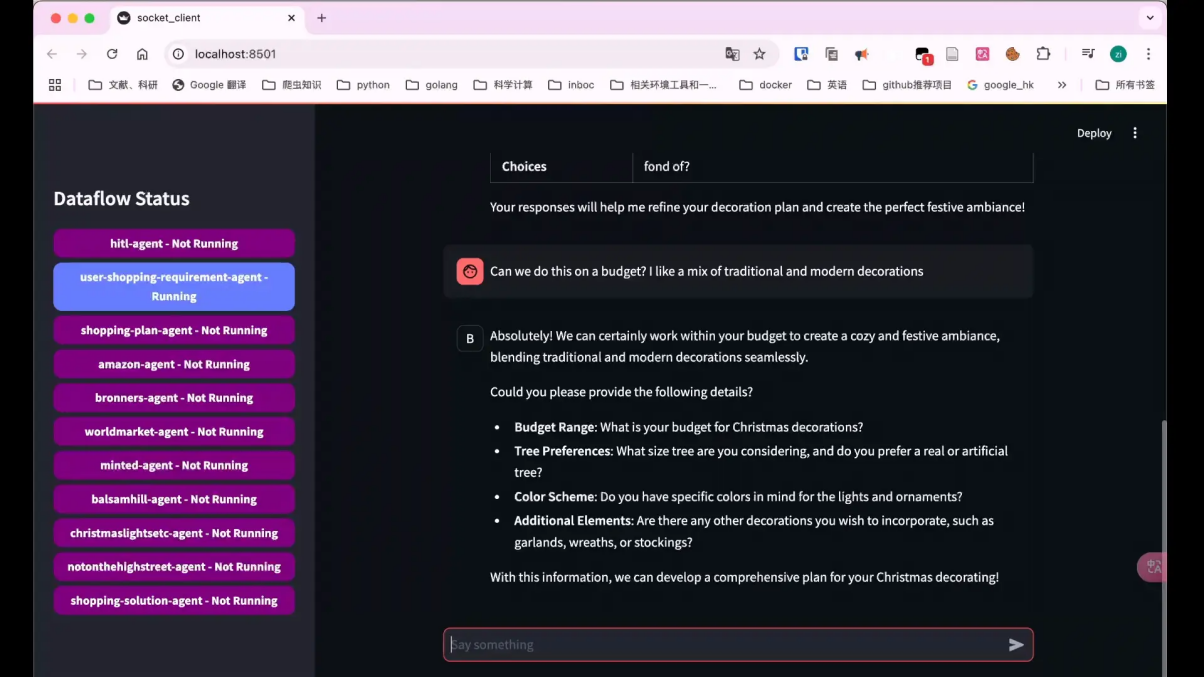
"Hey, I'm thinking about decorating my home for Christmas. I want it to feel cozy with lights, a nice tree, and some fun ornaments."



The system responds as above, expecting you to provide further details to offer better recommendations.

**Step 2: Additional Requirements**

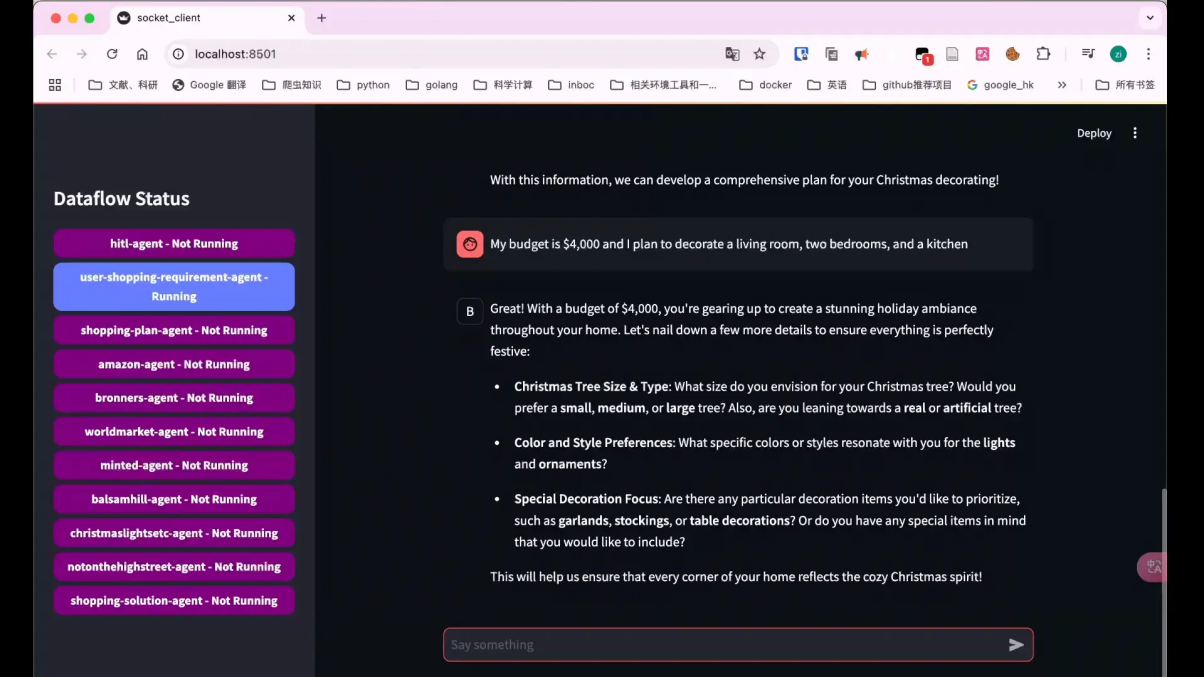
"Can we do this on a budget? I like a mix of traditional and modern decorations."



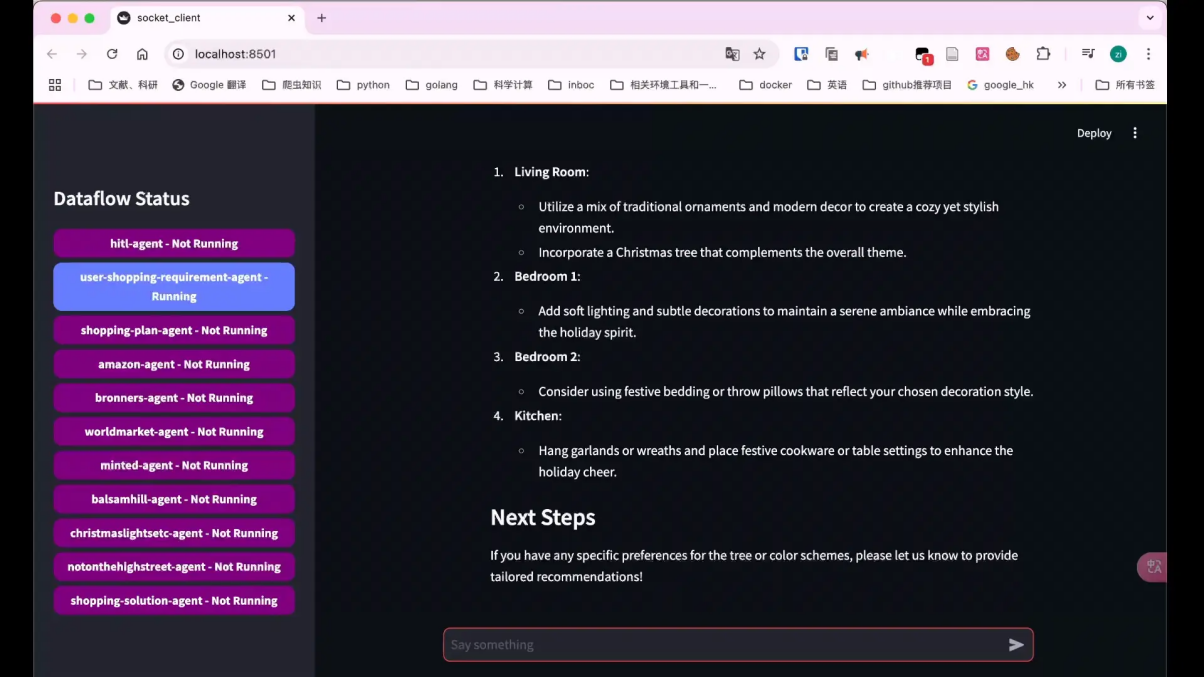
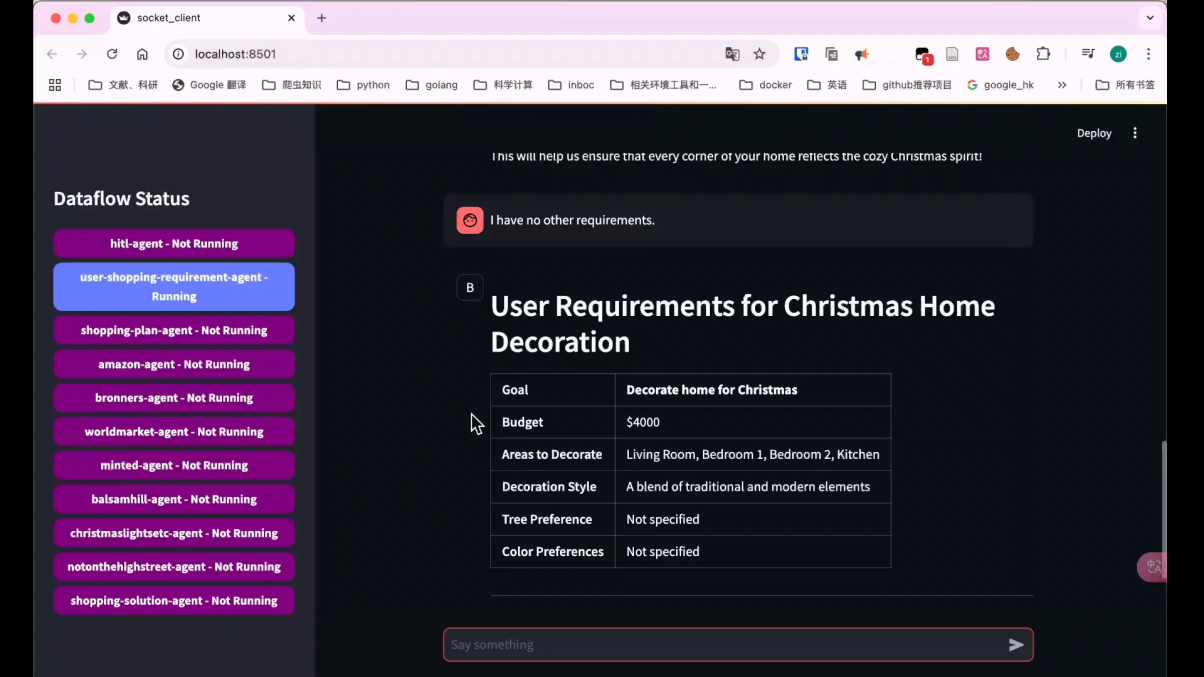
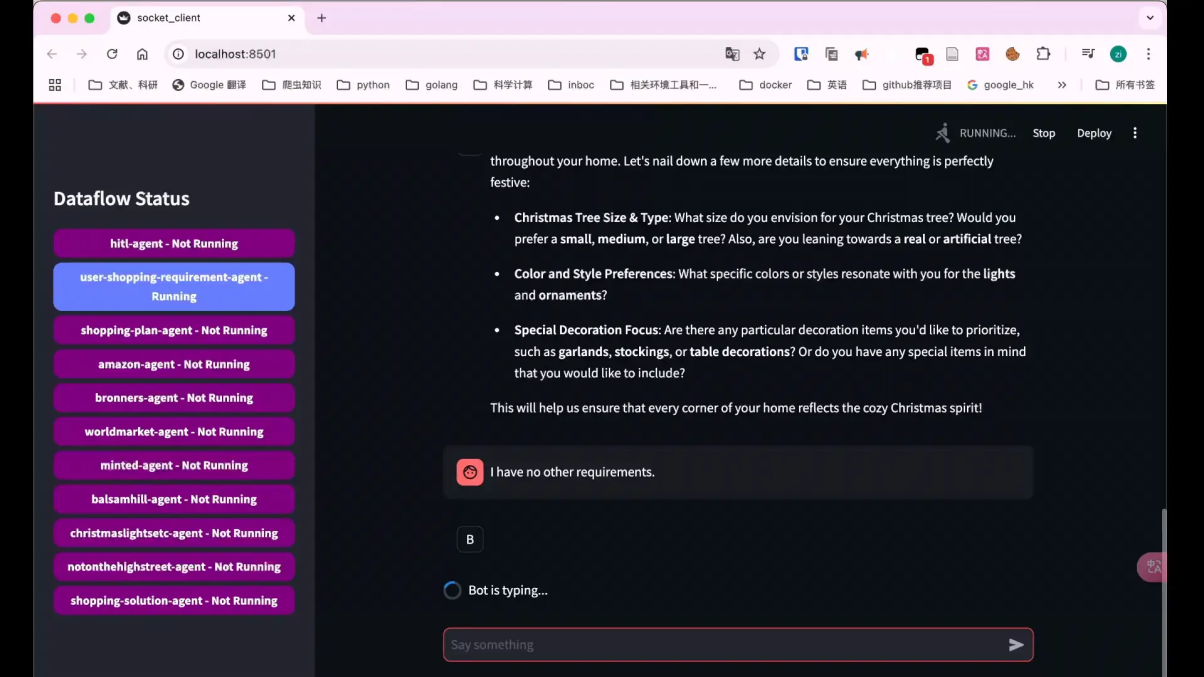
Continue refining your needs. The system will guide you through your requirements, so feel free to express yourself.

**Step 3: Generate Shopping Plan**

"Can you put together a list of things I should get? If you have any cute DIY ideas, I'd love to see those too!"

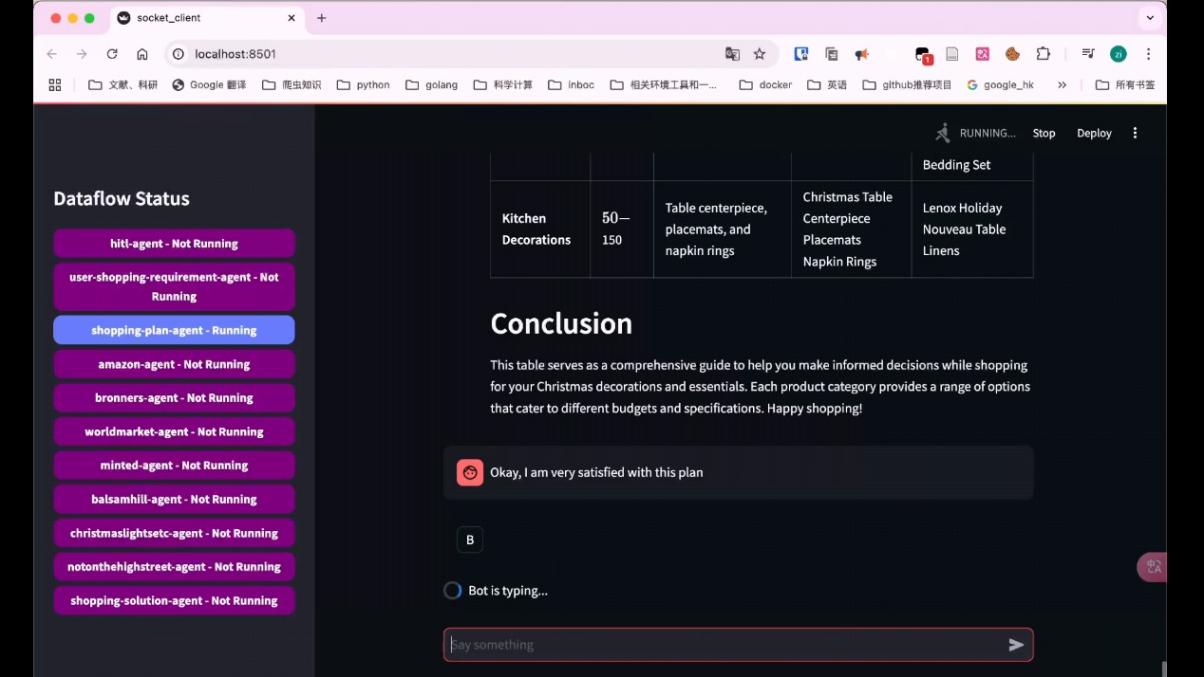
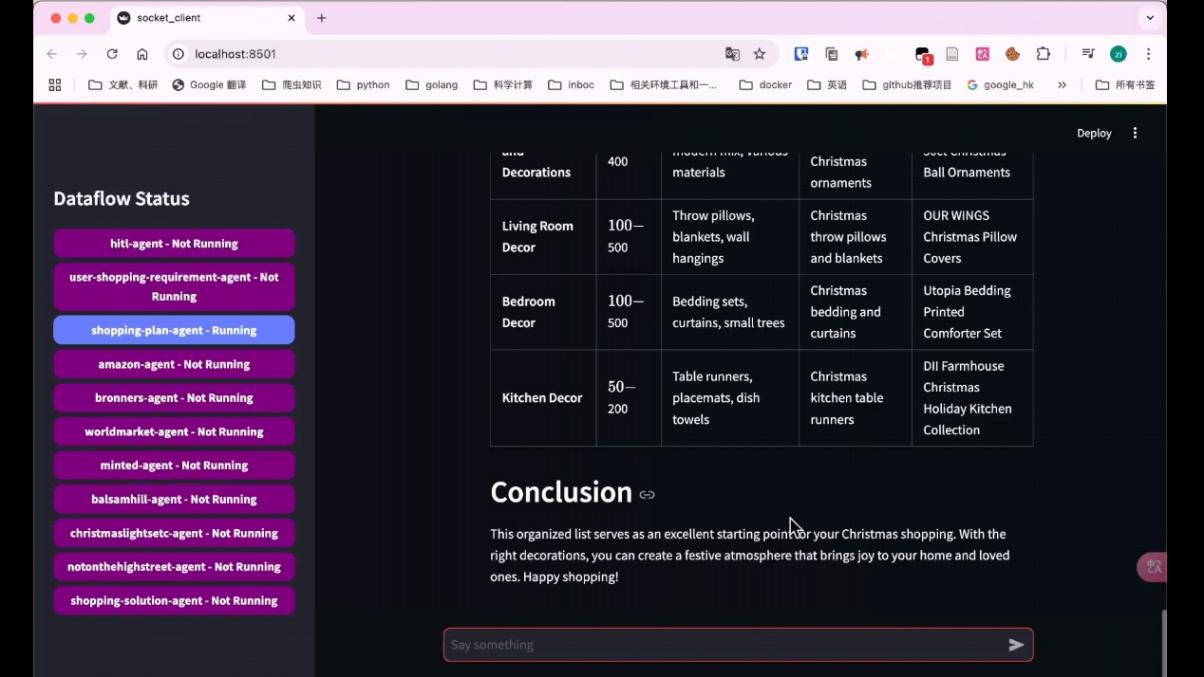
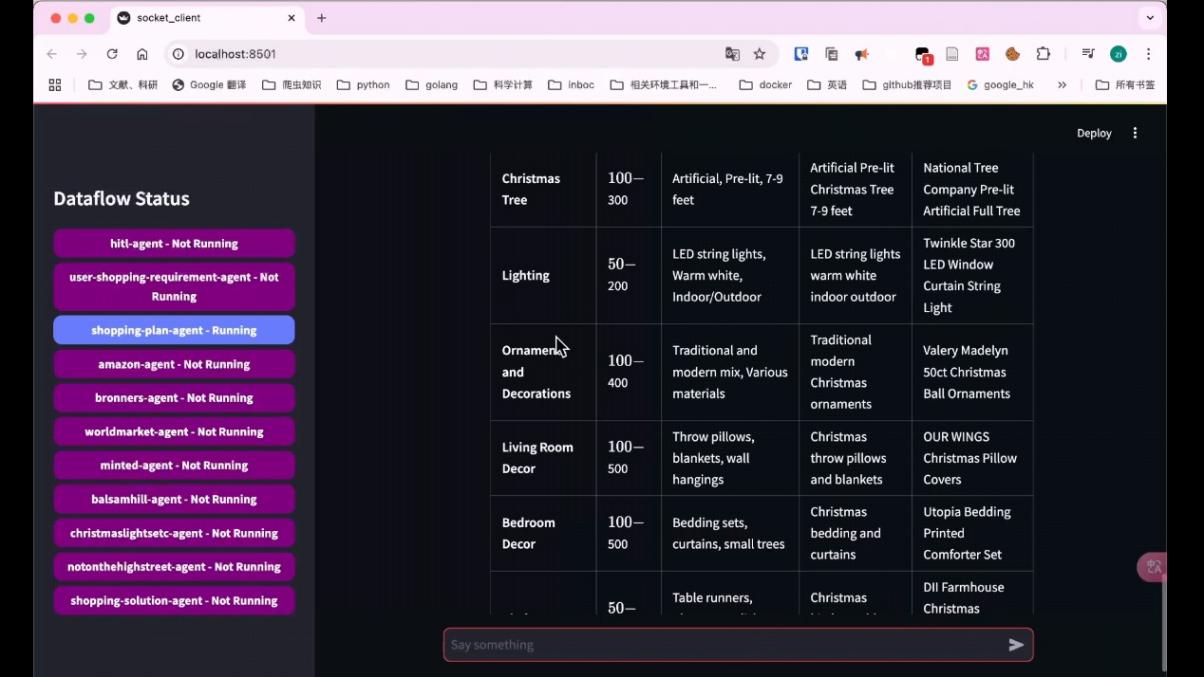
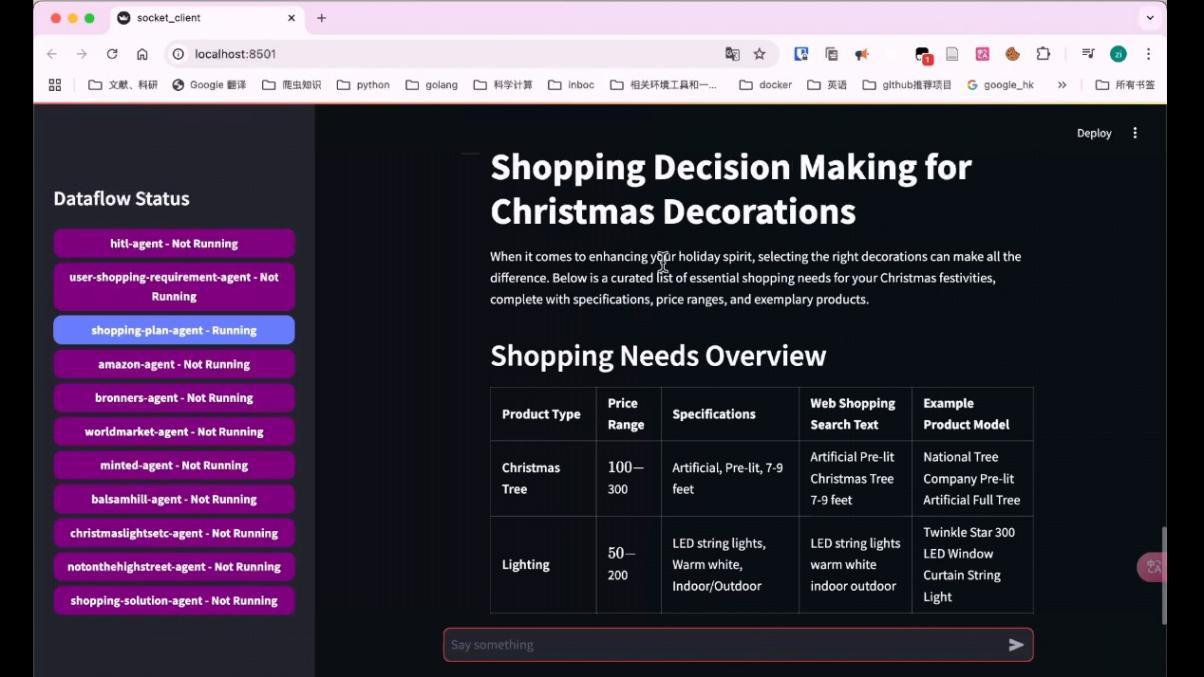
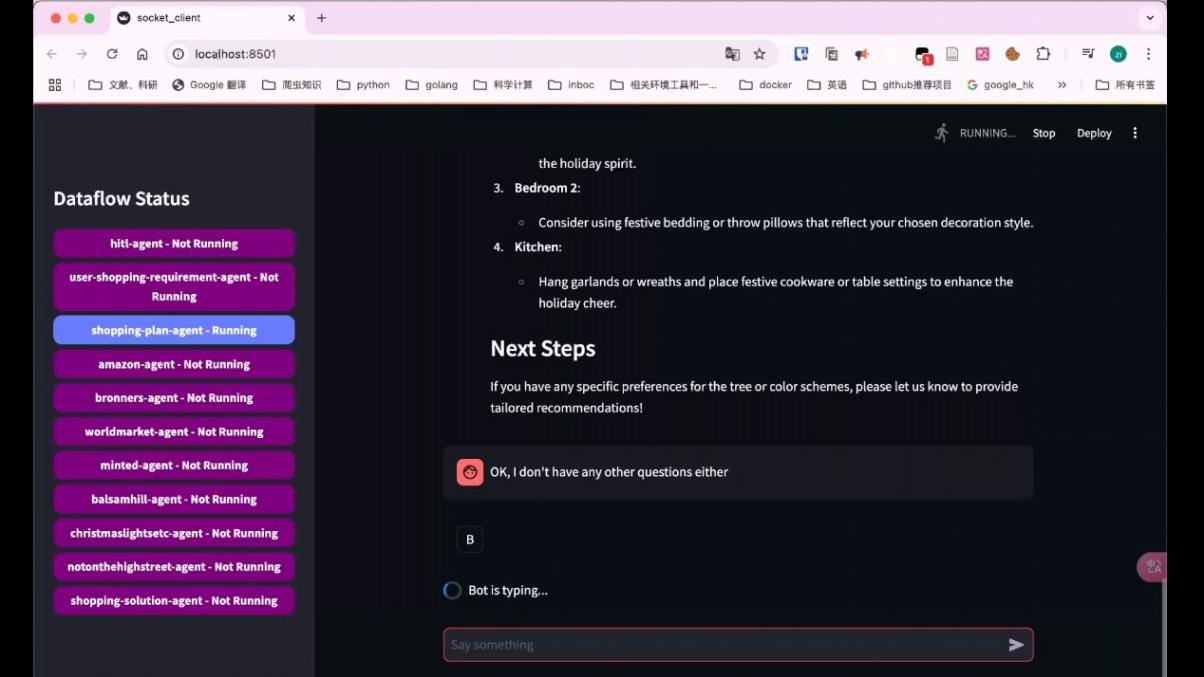


At this point, a rough plan is completed. The system will continue to ask for fine-tuning.



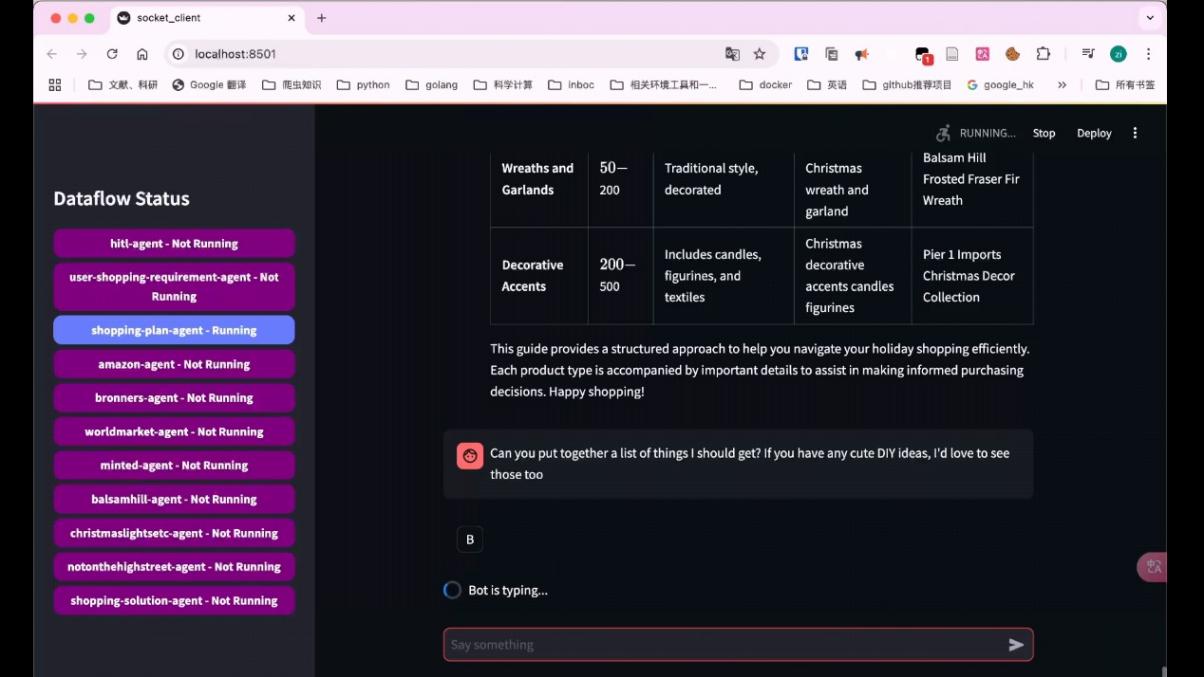
**Step 4: Fine-tune the Shopping Plan**

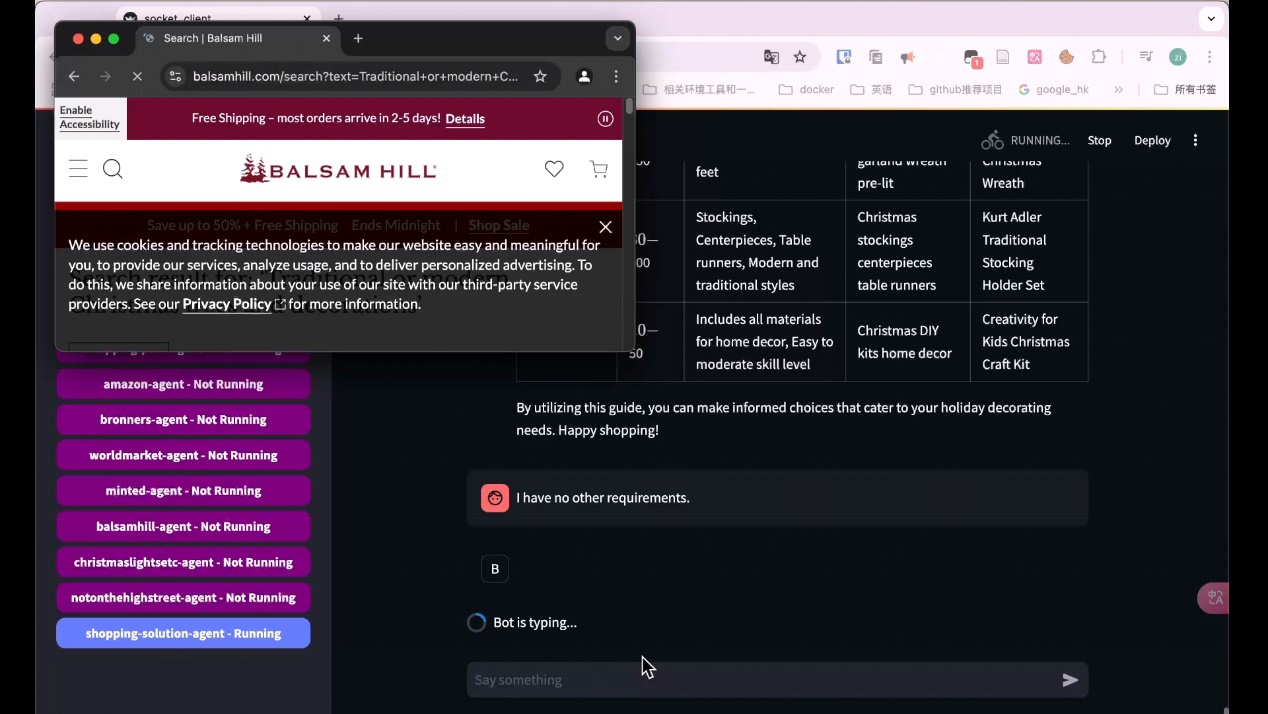
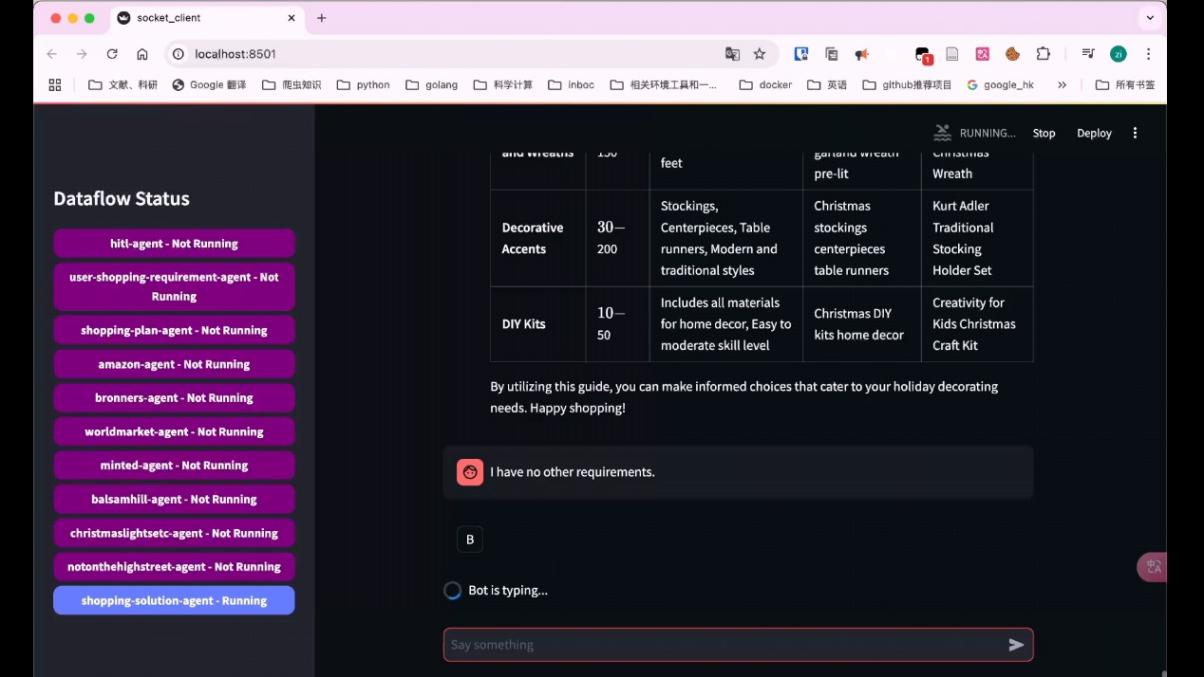
"These ideas are awesome! I love the wreath suggestion. Can you also add some nice dinnerware for our Christmas dinner to the list?"



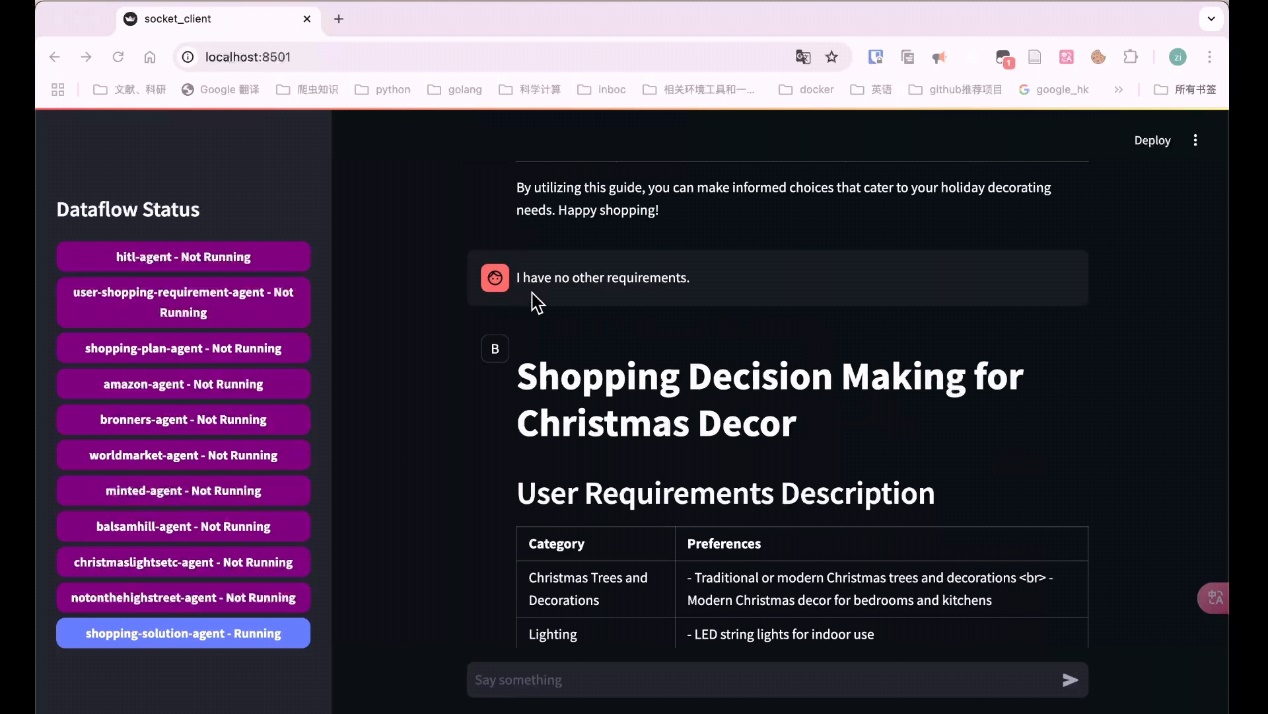
**Step 5: Provide Product Recommendations**

"Okay, I am very satisfied with this plan." I have no other requirements.



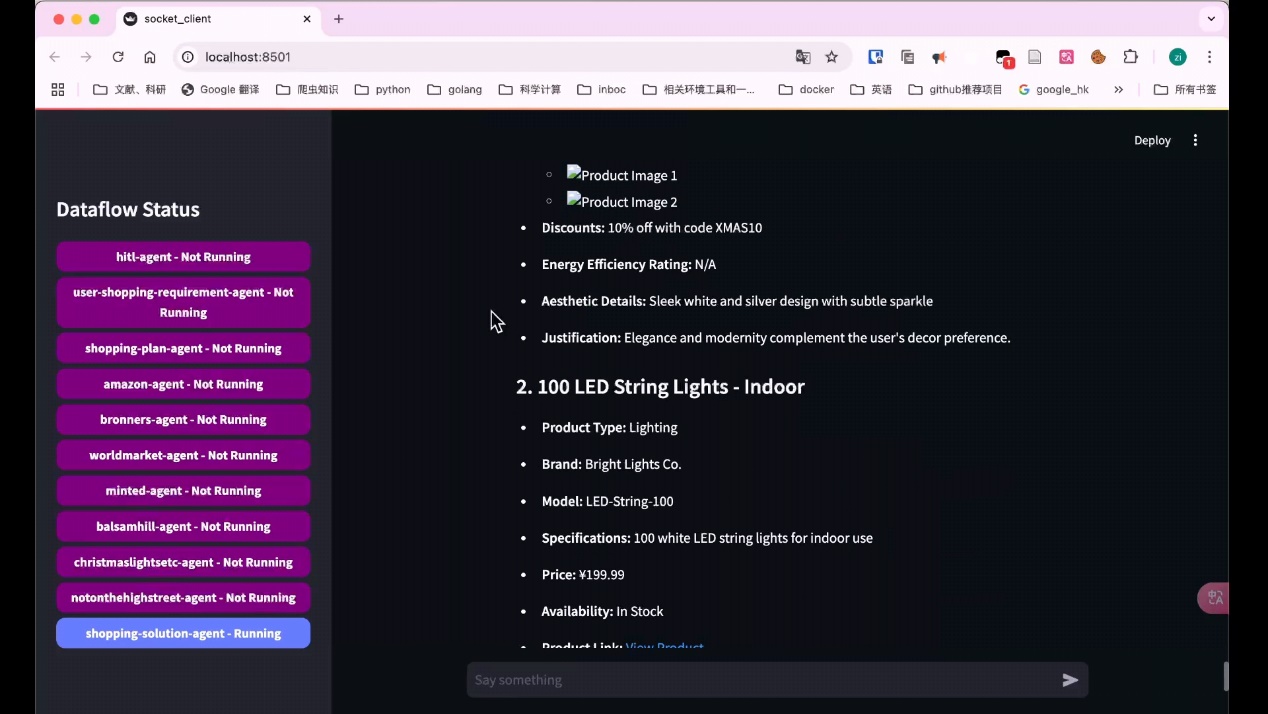
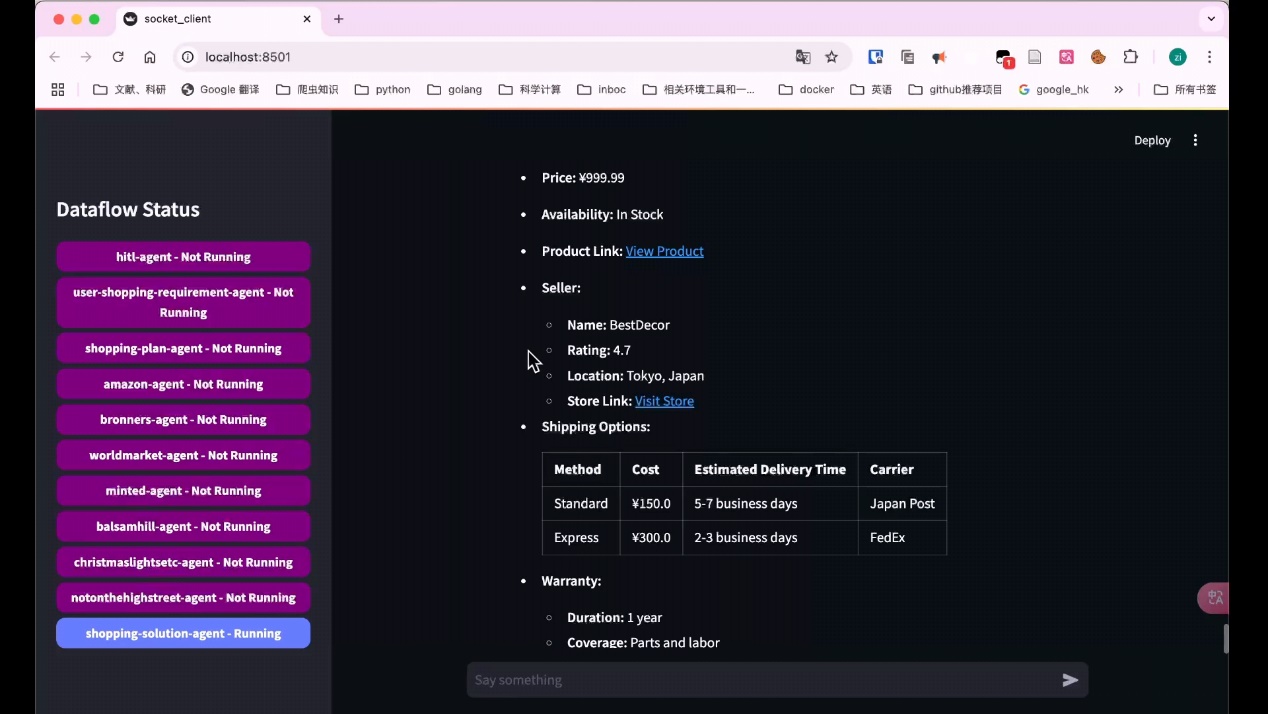
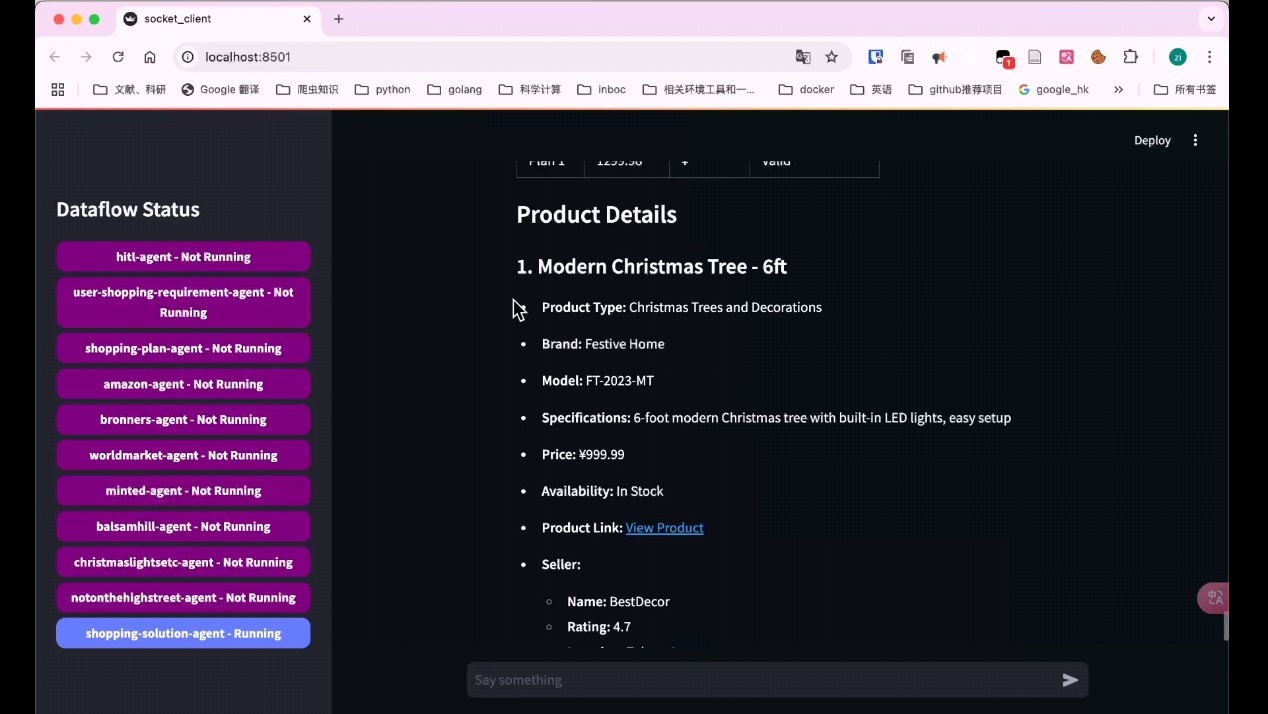
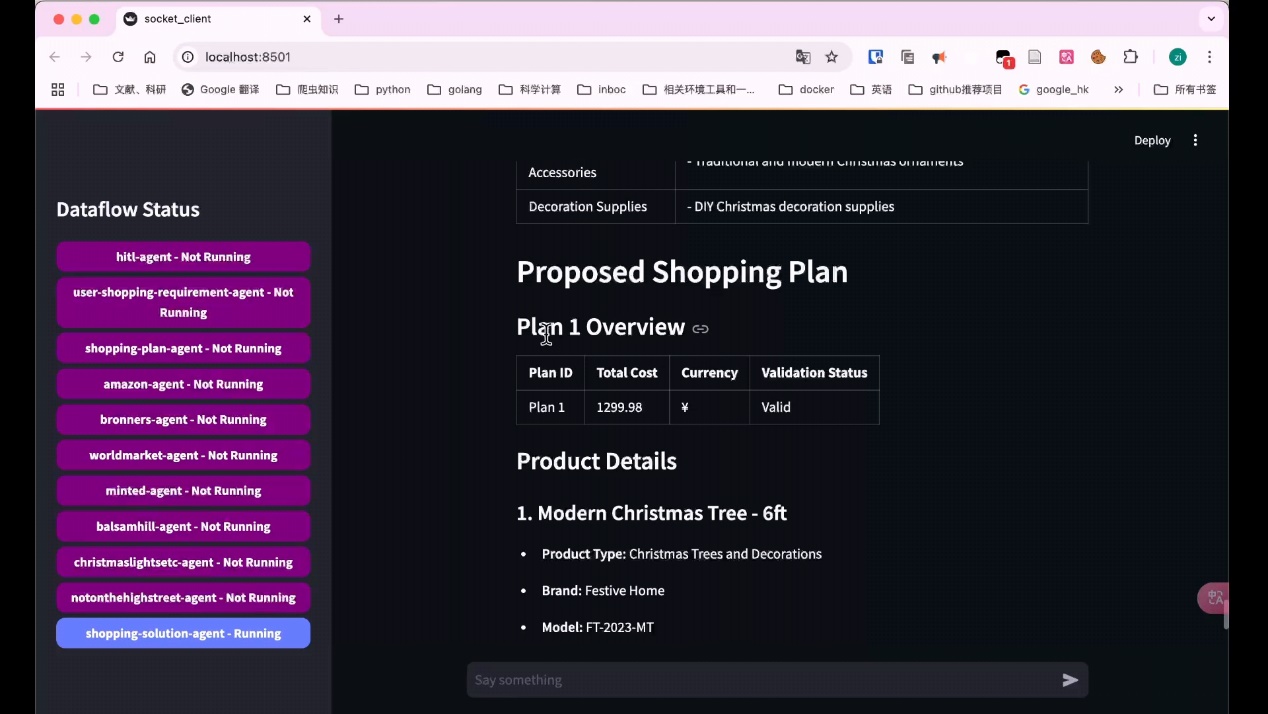


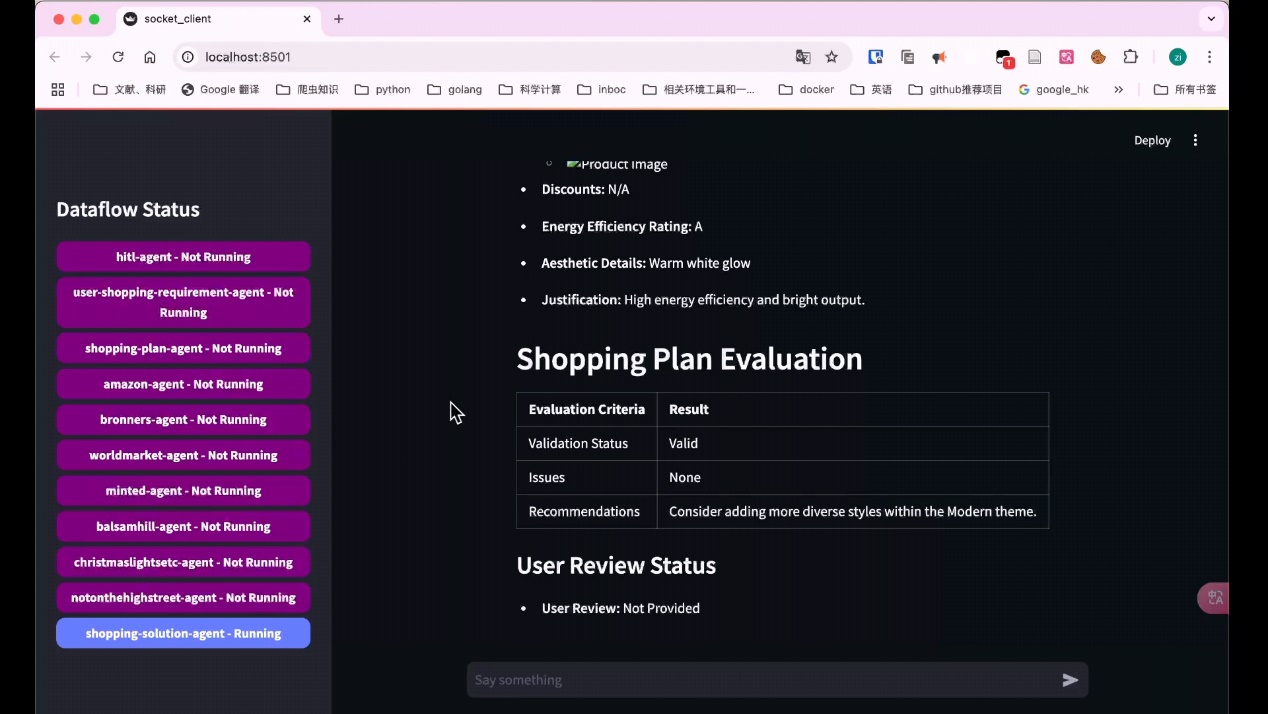
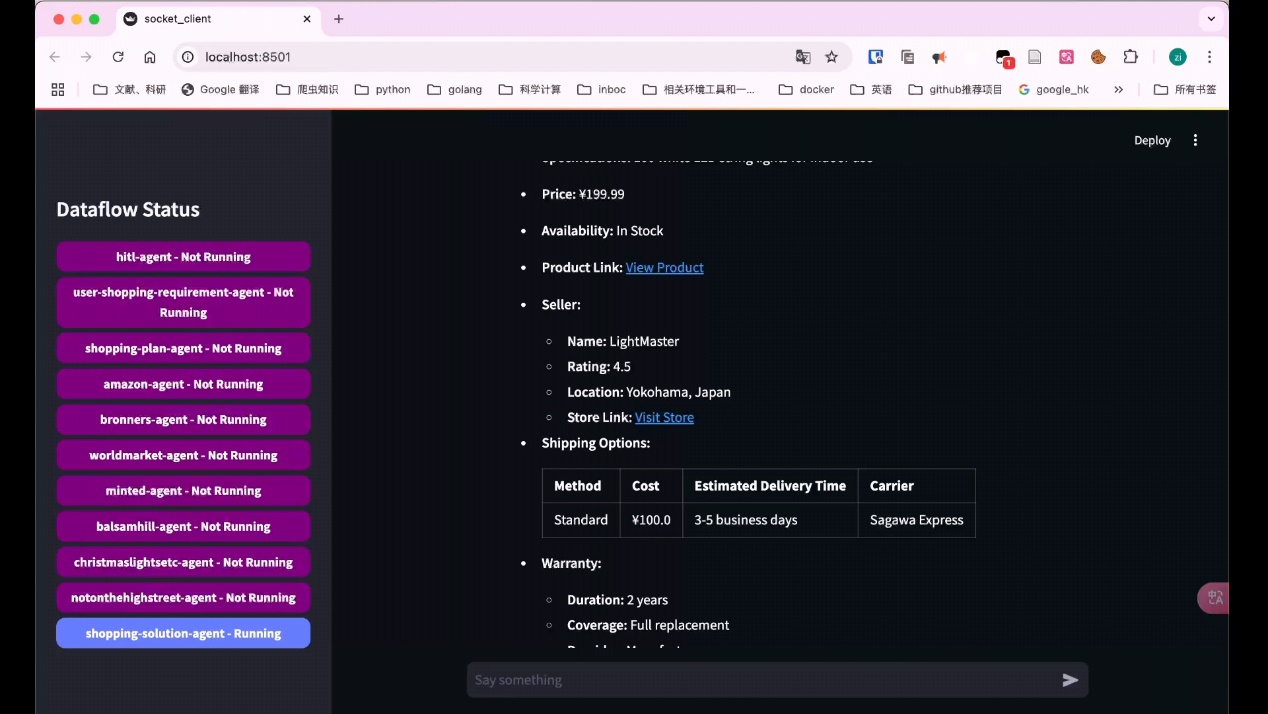
The system will start retrieving data from shopping websites to search for items in your plan. The data status on the left will update to the result search phase, which is the shopping-solution-agent.



Due to varying network speeds in different regions, the system's response time may take around 1-2 minutes. Once the search is complete, the system will provide a final report as shown in the example.

**Note**: Our item search agent is based on Selenium, so results may be affected by webpage changes or network conditions. It’s normal for the agent to sometimes find no items, and results may not be consistent every time.



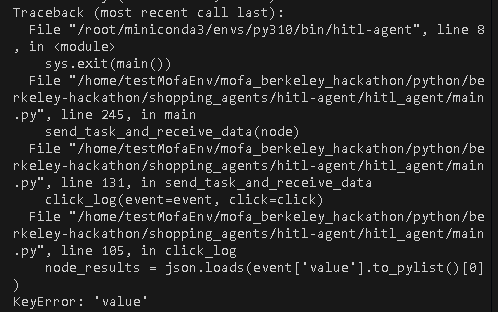
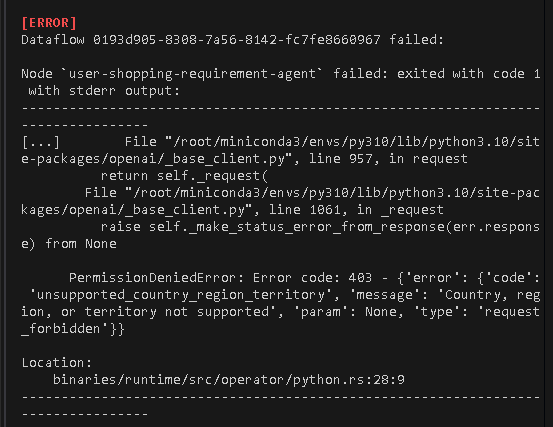


The system will ultimately provide a comprehensive shopping suggestion, including multiple items. Each item’s type, brand, model, features, price, and discounts will be summarized for you. Finally, an evaluation of the shopping plan will be provided to help you better adjust your plan.

### 5 Common Issues

**5.1 Troubleshooting**

a. Errors due to incorrect proxy node settings causing access denial or slow network speeds, or connection errors due to node issues.



**Solution**: Change the proxy node to a U.S. node, even if the latency is higher than other Asian or European nodes.

b. **Update Guide**

**Front-end Dataflow Status Not Updating**

1.The agent-node did not start successfully. Check the logs located in out/dataflow\_id/\*.log to identify any node startup errors.

2. If the agent-node has completed its tasks according to the logs but the status is not reflected on the page, there may be an issue with the page display or data transmission. In this case, please contact us by submitting an issue on GitHub Issues. We will respond promptly to address your concerns.

**c**. **How to View Running Logs**

1.After starting the system with dora-dataflow, an out directory will be created in your current directory.

2. If the agent-node has completed its tasks according to the logs but the status is not reflected on the page, there may be an issue with the page display or data transmission. In this case, please contact us by submitting an issue on GitHub Issues. We will respond promptly to address your concerns.

### 6. Maintenance and Updates

**6.1 Update Guide**

Run the following commands to fetch the latest code before running the backend:

git fetch origin

git pull origin develop

Our branch is develop. If you switch branches, replace with the branch name you want to pull.

### 7. Support and Feedback

**Technical Support**: If you have any questions about the code or anything else, feel free to contact our technical team:

**Feedback Channels**: You can submit a pull request on GitHub or use the contact information above to communicate your suggestions or feedback to our technical team.